Programación dinámica

La programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de sub-problemas superpuestos y subestructuras óptimas, como se describe a continuación. El matemático Richard Bellman inventó la programación dinámica en 1953 que se utiliza para optimizar problemas complejos que pueden ser discretizados y secuencializados.

La programación dinámica resuelve los subprogramas una sola vez y guarda sus soluciones en una tabla para su futura utilización; basándose en que "en una secuencia de decisiones óptima toda subsecuencia ha de ser también óptima".

Para que un problema pueda ser abordado por esta técnica ha de cumplir dos condiciones:

- La solución al problema ha de ser alcanzada a través de una secuencia de decisiones, una en cada etapa.
- Dicha secuencia de decisiones ha de cumplir el principio de óptimo.

Técnica cuantitativa que permite encontrar las decisiones óptimas en un proceso dividido en fases

Elementos que definen un problema de programación dinámica son:

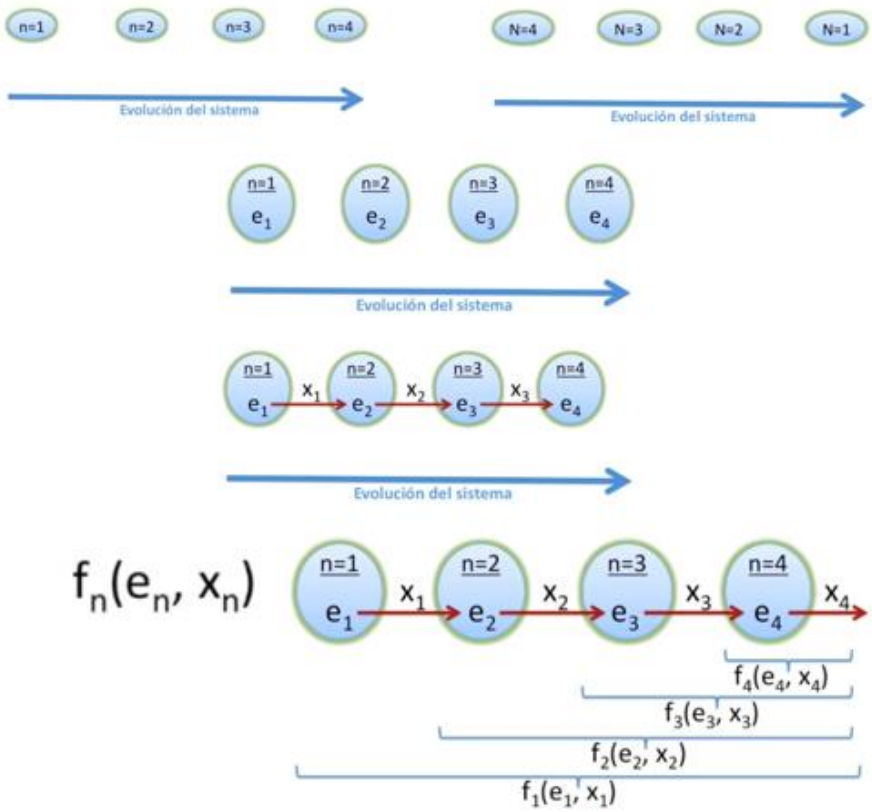
***Etapas *Estados *Variables de decisión *Función recurrente o recursiva**

Etapas: Son el periodo de tiempo, el lugar, el contexto, la fase o situación en donde se produce un cambio debido a una decision

Estados: Los estados muestran la situación actual del sistema cuando nos encontramos en la etapa n.

Variables de decisión: hacen referencia a toma de decisiones que se produce en una etapa y que provoca un cambio en el estado actual del sistema.

Función recurrente: La función de recurrencia refleja el comportamiento del sistema en función de los estados y las variables de decisión.



La programación dinámica es utilizada en compiladores, que consiste en solucionar cierto problema dividiéndolo en subproblemas más sencillos, calculando sus resultados y almacenándolos. Estos resultados posteriormente se utilizan para la resolución del problema final.
Almacenar resultados de subproblemas es una gran ventaja en cálculos dónde se repiten las mismas operación múltiples veces, mediante el método de la programación dinámica estas operaciones sólo se realizan una vez y se guarda la solución.
Se dice de la programación dinámica que es un método para resolver problemas que exhiben propiedades de problemas sobrepuestos y estructura óptima.

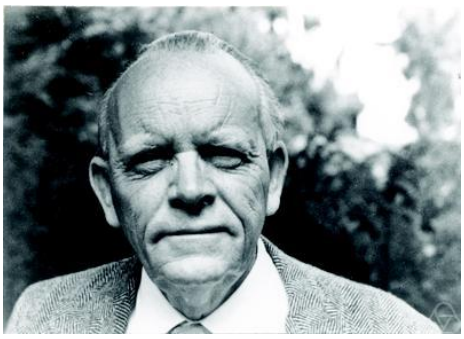
En grandes líneas, el diseño de un algoritmo de Programación Dinámica consta de los siguientes pasos:

1. ***Planteamiento de la solución como una sucesión de decisiones y verificación de que ésta cumple el principio de óptimo.***
2. ***Definición recursiva de la solución.***
3. ***Cálculo del valor de la solución óptima mediante una tabla en donde se almacenan soluciones a problemas parciales para reutilizar los cálculos.***
4. ***Construcción de la solución óptima haciendo uso de la información contenida en la tabla anterior.***

En general, los algoritmos obtenidos mediante la aplicación de esta técnica consiguen tener complejidades (espacio y tiempo) bastante razonables, pero debemos evitar que el tratar de obtener una **complejidad temporal** de orden polinómico conduzca a una **complejidad espacial** demasiado elevada
La Programación Dinámica se adapta a problemas de carácter secuencial, por ejemplo:

- Búsqueda de caminos más cortos entre dos puntos
- Planificación de tareas
- Gestión de recursos
- Gestión de stock

Algoritmo de vuelta atrás (BackTracking)



Vuelta atrás, (Backtracking) es una estrategia para encontrar soluciones a problemas que satisfacen restricciones. El término "backtrack" fue acuñado por primera vez por el matemático estadounidense D. H. Lehmer en la década de 1950.
El retroceso o vuelta atrás es una técnica de resolución de problemas que ***realiza una búsqueda exhaustiva, sistemática y organizada sobre el espacio de búsqueda del problema***, aplicable a problemas de optimización, juegos, búsquedas entre otros.
Se llaman algoritmos de vuelta atrás, porque en el caso de no encontrar una solución a una sub-tarea se retrocede a la sub-tarea anterior y se prueba otro camino diferente a los probados anteriormente.

Los problemas que manejan los algoritmos exhaustivos, se caracterizan por:

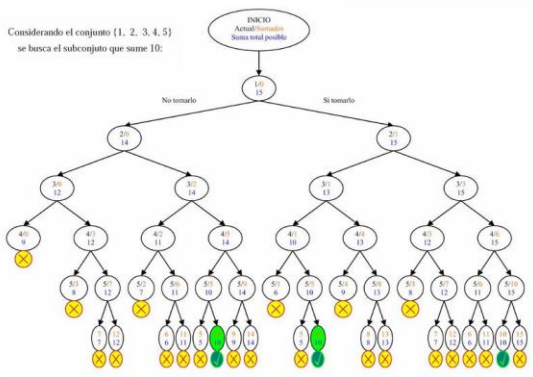
a. **Se trata generalmente de problemas de optimización, con o sin restricciones.**
b. **La solución es expresable en forma de una secuencia de decisiones**.
c. **Existe una función denominada factible que permite averiguar si en una secuencia de decisiones, la solución en curso, viola o no las restricciones.**
d. **Existe una función, denominada solución, que permite determinar si una secuencia de decisiones factible es solución al problema planteado.**

El esquema general de solución presenta los siguientes pasos:

a. **Vuelta Atrás hace un recorrido en profundidad del espacio de búsqueda partiendo de la raíz.**

b. **El recorrido en profundidad regresa sobre sus pasos, retrocede, cada vez que encuentra un camino que se ha acabado o por el que no puede continuar.**

c. **En un recorrido en profundidad o en anchura sobre un espacio de búsqueda se conoce de antemano el orden en que se van a generar o recorrer, sus nodos**.



Este tipo de metodología es muy genérica, ya que trata de recorrer ***todas las posibles soluciones*** y ***encontrar la más apropiada*** para cada caso o las soluciones que satisfacen una condición, con lo que puede ser aplicada en la mayoría de problemas. Por ello, esta técnica de resolución es usada en muchos ámbitos de la programación, por ejemplo, para el cálculo de expresiones regulares o para tareas de reconocimiento de texto y de sintaxis de lenguajes regulares; también esta técnica es usada en diversos lenguajes de programación tales como ***Planner y Prolog***. Además, se usa en los análisis sintácticos de los compiladores. Su uso en inteligencia artificial ha sido muy importante, dando lugar a nuevos tipos de búsquedas como el A estrella.

Ejemplos de problemas comunes resueltos usando Vuelta Atrás

* Sudoku
* Problema de los movimientos de un caballo
* Las ocho reinas